Future Generation Computer Systems 86 (2018) 1193–1208

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An efficient data delivery mechanism for AUV-based Ad hoc UASNs

Gour Karmakar [a,*], Joarder Kamruzzaman [a,b], Nusrat Nowsheen [b]

[a] School of Engineering and IT, Faculty of Science, Federation University, Australia
[b] Faculty of Information Technology, Monash University, Australia

HIGHLIGHTS

- For coordinated data collection, AUVs need to be controlled by another AUV commands.
- Currently, no data delivery protocol exits for a 3D UASN comprising only AUVs.
- A data forwarding technique is introduced by controlling node movement intelligently.
- The delivery protocol shows the outstanding performance in terms of many metrics.
- This work will contribute to the deployment of underwater Internet of Things.

ARTICLE INFO

Article history:
Received 28 February 2017
Received in revised form 17 September 2017
Accepted 14 October 2017
Available online 22 November 2017

Keywords:
Data delivery protocol
Mobile ad hoc underwater network
3D underwater acoustic sensor networks
Autonomous underwater vehicles

ABSTRACT

Existing 3D Underwater Acoustic Sensor Networks (UASNs) are either fixed having nodes anchored with the seabed or a combination of Autonomous Underwater Vehicles (AUVs) and a fixed UASN where AUVs are controlled to move along paths for data collection. Existing data delivery protocols for such AUV equipped networks are designed in a way where AUVs act as message ferries. These UASNs are deployed for a specific service such as asset (e.g., oil pipes, shipwreck) monitoring and event detection. For a coordinated data collection, to deploy a network for any service like information discovery in an ad hoc manner, it requires a 3D UASN consisting of only AUVs and the movement of an AUV needs to be controlled by another AUV through commands. To our knowledge, no such data delivery protocol required for a 3D UASN comprising only AUVs exists in the current literature that can efficiently handle data collection and delivery. To address this research gap, in this paper, an AUV-based technique for ad hoc underwater network, namely AUV-based Data Delivery Protocol (ADDP), is introduced which ensures data delivery within a given time-constraint by controlling node (i.e., AUV) movement at each hop through commands of a node. The performance of the proposed protocol has also been evaluated and compared with existing relevant protocols in terms of packet delivery ratio, routing overhead and energy consumption considering various network scenarios and sizes. Results exhibit outstanding performance improvement achieved by the proposed protocol for all metrics.

1. Introduction

Recent times have seen significant interest of deploying underwater sensor networks in diverse applications like oceanographic data collection, underwater exploration, surveillance, marine conservation and many others. The collected data are then delivered to on-shore nodes connected to the Internet, enabling an Internet of Things (IoT) application. These underwater sensor networks mainly use acoustic communication, rather than radio or optical communication. Though low frequency (30–300 Hz) transmission can propagate significant distance through conductive sea water, but requires large antenna and high transmission power, while high frequency radio transmission is severely limited by low propagation distance. Optical transmission is possible but high precision is needed in pointing the narrow laser beams. Acoustic signals are less affected by the turbidity and particles suspended in water and able to reach large distances (over 20 km [1]). Thus it is the only practical and feasible method for underwater communications. However, due to the disruptive nature of the communication medium, large propagation delay of acoustic signal and mobility of sensor nodes, reliable data delivery for Underwater Acoustic Sensor Networks (UASNs) is difficult. Many existing protocols rely

* Corresponding author.
E-mail addresses: Gour.Karmakar@federation.edu.au (G. Karmakar), Joarder.Kamruzzaman@federation.edu.au (J. Kamruzzaman), n_nowsheen7@yahoo.com (N. Nowsheen).

https://doi.org/10.1016/j.future.2017.10.025
0167-739X/

on fixed communication architecture which limit their applicability to achieve reliable data delivery in different application and network scenarios. Existing underwater protocols [2–6] depend mainly on flooding-based techniques for improving data delivery. This introduces redundancy in the network, and consequently incurs large communication overhead and energy consumption in the resource constrained UASNs. To improve energy efficiency and the data delivery reliability of underwater sensor networks, existing works have considered parameters such as location, depth, energy information of sensor nodes, link quality, node movement and traffic flow. On the other hand, with the recent advances of technology, continued development of new AUVs increases the range of applications of UASN. Thus, it makes an AUV-based architecture suitable for mission critical applications, such as, surveillance, search/rescue operation, seismic monitoring and oil spill detection [7–9]. Unlike periodic ocean monitoring applications which can tolerate long network delay, these applications require frequent data collection and offloading of those data to an on-shore base station maintaining short delay tolerance. Data are considered useful only if they reach the surface before their lifetime expires.

In the above situations, an ad hoc network consisting of only AUVs is more feasible. The network in question can be deployed instantly to react to sudden changes in the environment and deliver data within a given deadline. To the best of our knowledge, no protocol has been developed for ad hoc network where movement of nodes (AUVs) is controlled by another node through commands. Controlling node movement using commands will enable applications to cover a large geographic area by employing a small number of nodes and save energy by allowing short distance data communication. This motivates us to deploy an UASN in an ad hoc manner using AUVs in any deployment context to collect data in a cost-effective way. Consequently, the other aspect of our research targets to develop a data delivery protocol to facilitate AUV-based underwater ad hoc network which can provide any service from new information discovery to event monitoring and does not require any fixed WSN to be installed in the data collecting region. Moreover, due to the dynamic nature of acoustic link, data forwarding decision at each hop is more suitable than end-to-end route discovery. In this regard, forwarding decisions at each hop should take place by observing the surrounding environment, such as link statistics, traffic flow and node movement. Therefore, in this paper, to address this research issue, we present an *AUV-based Data Delivery Protocol* (ADDP) to explore the ocean environment by forming an ad hoc network which employs a group of AUVs. The AUVs begin their missions within the area where an event (e.g., wrecked ship/aeroplane) takes place. However, deployment of multiple AUVs to achieve a mission goal in ocean environment requires the AUVs to behave in a cooperative manner. The concept of ad hoc network here is based upon the formation of communication links among mobile vehicles towards achieving a common mission goal.

AUVs move autonomously in the ocean. Their movement can also be controlled by *command* and *response* message exchanges. In our protocol, we exploit this feature to improve data delivery by providing communication between two nodes even if they are not initially located within each other's data communication range. The reliability is ensured by delivering data in a timely manner which is achieved by command-based node movement to bring nodes within each other's communication range, if required. To report intermediate findings during a mission, data collected by an AUV need to be sent to the surface station or IoT node for analysis. To achieve this, AUVs are scheduled to surface at different times during their mission. The contributions of this paper are summarized as follows:

(i) A novel AUV-based technique for ad hoc underwater network, named AUV-based Data Delivery Protocol (ADDP), is introduced which ensures data delivery within a given time-constraint by controlling node movement at each hop through commands.
(ii) In ADDP, a time-scheduled operation is maintained where AUVs surface at different times during a mission at a fixed interval and all data are carried by an AUV, called target node, whose schedule meets delivery deadline. A forced movement technique is incorporated to move either sender or receiver or both AUVs so that they come within each other's communication range, leading to the target node. This significantly improves data delivery success ratio.
(iii) Simulation models of the proposed protocol has been developed considering the characteristics of an underwater acoustic environment using network simulator NS2. Its performance has also been evaluated and compared with existing relevant protocols in terms of *packet delivery ratio* (PDR), routing overhead and energy consumption considering a number of different network scenarios and sizes.

The rest of this paper is organized as follows. The related works are reviewed in Section 2. We present the system model in Section 3. The network initialization phase is discussed in Section 4. A brief overview of the proposed data delivery protocol is covered in Section 5. A detailed description on how this technique works, as well as how it has been used to improve the reliability in data delivery for underwater mission critical applications is presented in Section 6. Section 7 shows a state diagram to describe the different states of an AUV during protocol operation. Section 8 provides the experimental results and finally, we conclude this work in Section 9.

2. Related works

Over the last few years, significant advances have been made in the development of data delivery protocols for UASNs. With the development of such protocols, the scope of their applications continues to grow. This, in turn, implies that the performance of the protocols needs to improve. Both commercial and military applications are now calling for data delivery among sensor nodes and AUVs in network scenarios. This demand has moved current researchers towards the development of reliable data delivery techniques for UASNs. Many of the existing studies [2,3,5,6,10–12] assume fixed communication structure where the sink node is deployed at the surface and its location is known by other nodes in the network before deployment. However, these assumptions limit their applicability and flexibility. Existing underwater data delivery protocols can be broadly categorized as: (a) Location-Unaware and (b) Location-Aware protocols.

2.1. Location-unaware data delivery protocols

Protocols in this category do not require 3D location information of sensor nodes to be known for delivering data packets. Most of these protocols including *Depth-Based Routing* (DBR) [5], *Depth-Based Multihop Routing* (DBMR) [13] and *hydraulic pressure based routing* [14] use depth information to forward data packets towards the sink node. In depth-based protocols [5,13,15,14], data packets are forwarded from higher to lower depth nodes based on depth information of sensor nodes until the sink node is reached. However, depth-based protocols are not applicable to horizontal networks where nodes are deployed at the same depth. Another group of location-unaware protocols [16,17] have been developed based on the clustering technique for data forwarding. Even though they are energy efficient, the structure of the cluster is affected in

underwater due to node movement. Some protocols [18,19] have been proposed for delay-tolerant underwater network architecture, however, like traditional DTN protocols, they also did not consider the inherent disruptive nature of the acoustic link during data forwarding.

2.2. Location-aware data delivery protocols

Most studies found in the literature base their works on the active localization techniques. Location-aware protocols [2–4,6] forward data packets based on the 3D location of sensor nodes. Data are forwarded until the sink node is reached. However, the success of the protocols depends on localization technique to estimate accurate locations of sensor nodes because *Global Positioning System* (GPS) waves cannot propagate through water. Even though there exist some localization techniques in underwater [20–24], yet localization is a difficult task to accomplish given the inherent mobility of underwater nodes due to ocean current. Most existing underwater localization algorithms require reference nodes at known locations to assist localization of other nodes. However, location assumptions can only be applied to the nodes anchored with the ocean bottom as they know their locations during deployment time or to AUVs which are equipped with internal navigation system [25]. In addition, the ocean current creates coverage holes even in 3D fixed networks where nodes are fastened with the seabed.

An energy efficient routing protocol based on mobicast routing technique for UWSNs comprising static sensors anchored with the seabed is introduced to reduce the impact of the coverage holes on data gathering [26]. Dara are collected from the sensor nodes by an AUV traveling on a path defined *a priori*. The main loophole of this protocol is that it has created an optimization problem on how to determine paths and their number that can be used to collect data from all or the maximum number of sensor nodes. To address this problem, for UWSNs, an opportunistic multi-hop data forwarding technique incorporating the DTN concept is introduced in [27]. Here, some of the sensor nodes anchored with the seabed are selected by message ferry as *gateway* (GW) notes before data collection considering their residual energy and using a technique which maximizes the life time of GWs. A message ferry is used in collecting data from GWs. Routing paths from GWs to any sensor nodes are established with a message broadcasting. The coverage hole problem is addressed by predicting the movement of a sensor node with the ocean current within a specific time window. Since GWs are selected before data collection, there is no guarantee that all GWs will always exist within the coverage of the message ferry traveling on the same path used in GWs selection and the traveling path of a message ferry is intuitively selected. The determination of the travel roadmap for an AUV, which can improve data collection from a number of selected underwater regions within a bounded time delay is presented in [28]. The travel roadmap is constructed considering the ocean currents, the motion of an AUV and the obstacles around the visiting regions. The techniques involved to develop a roadmap capable of avoiding the collision between an AUV and the obstacles or seabed are prize collecting traveling salesman problem and sampling-based motion planning. The main loophole of this approach is that the roadmap has been constructed for only an AUV. For multiple AUVs, the cooperative behaviors between AUVs to perform a specific mission task (e.g., monitoring an asset) is controlled by two predefined rules [29]. The movement paths of AUVs are generated using their locations cooperatively determined in collaboration with a network comprising a set of fixed sensor nodes installed on the seabed. This is developed to deliver a particular service such monitoring an asset but other important services like discovering new information required for marine community have not been considered. Since the movement is controlled by two rules, for cooperative task, one AUV cannot directly command others. The scheduling for surfacing to deliver the data to on-shore sink has not been incorporated. To collect the timely data from fixed sensor nodes and detect an event occurred, the number of AUVs and their surface scheduling are determined by minimizing the average data and the delay required for resurfacing [30]. The movement trajectories of AUVs are regarded as a piecewise set of line segments like oil pipes and they are produced using an extended Euler cyclic. This will not work for the AUVs moving on non-linear paths.

Data collection protocols developed so far serve either anchor-based architecture or a combination of AUV(s) and fixed nodes where the movement of AUVs is controlled by rules or trajectories. On the other hand, with the recent advances of technology, continued development of new AUVs will increase the range of applications. To the best of our knowledge and the literature presented in this paper, no protocol has been developed for deploying ad hoc network consisting of only AUVs to perform mission critical services (e.g., event detection, discovering new information, monitoring an asset) where movement of an AUV is controlled by another through commands in a cooperative manner. Controlling node movement using commands will enable applications to cover a large geographic area by employing a small number of vehicles and save energy by allowing short distance data communication.

3. System model

We consider a network of n AUVs to explore and collect data from the region of interest. As shown in Fig. 1, the total area is partitioned into a number of equal cubic sub-areas, each being assigned to one AUV represented by a number from 1 to n. Equal cubic sub-areas are chosen for the deployment of AUVs because cube can create 3D tessellation underwater yielding no void region within the monitoring area and it is easier to maneuverer an AUV keeping its movement within its allocated cubic sub-areas. It also make analysis of movement scheme and other related parameters more tractable. To report intermediate findings during a mission, AUVs are scheduled to surface at a fixed interval of T_i. At every T_i, an AUV i, where $i \in \{1, 2, \ldots, n\}$ goes to the surface and delivers its stored data packets to an on-shore station. For coordinated warfare comprising land, water and air, or some special event detection in sea, an on-shore station can be a wireless sink or Internet access point or an IoT object. After surfacing, each node is rescheduled and goes back to its pre-defined area to continue data collection. ADDP requires each AUV node to know the area map, i.e., the area assigned to each node and the surfacing schedule, i.e., the surfacing time of each node in the network. A surfacing time indicates when an AUV is going to surface next for offloading data to the onshore station. The surfacing schedule is determined prior to network deployment. After every surfacing time expires, each node is considered unavailable for a fixed duration of T_{unv}. This duration is utilized by the node to go to surface, offload data and get back to its original area. During this time, other nodes will not include this surfacing node for any of their protocol operations.

Each AUV calculates and attempts to deliver data within a data delivery deadline (t_c), i.e., the maximum time limit by which a packet should be delivered to another AUV which is going to surface next for offloading data. We denote the immediate next scheduled surfacing node as *target* node. As AUVs are scheduled to surface at different times during a mission, a packet is considered to be successfully received by an onshore station once it is delivered to the *target*. A node is selected as *target* if its surfacing time is less than the packet's t_c. Therefore, the proposed protocol searches for a *target* AUV which is able to deliver data packets and surface before its t_c expires. To determine t_c, each AUV needs to know the lifetime of a data packet (T_d). The lifetime of a data packet denotes

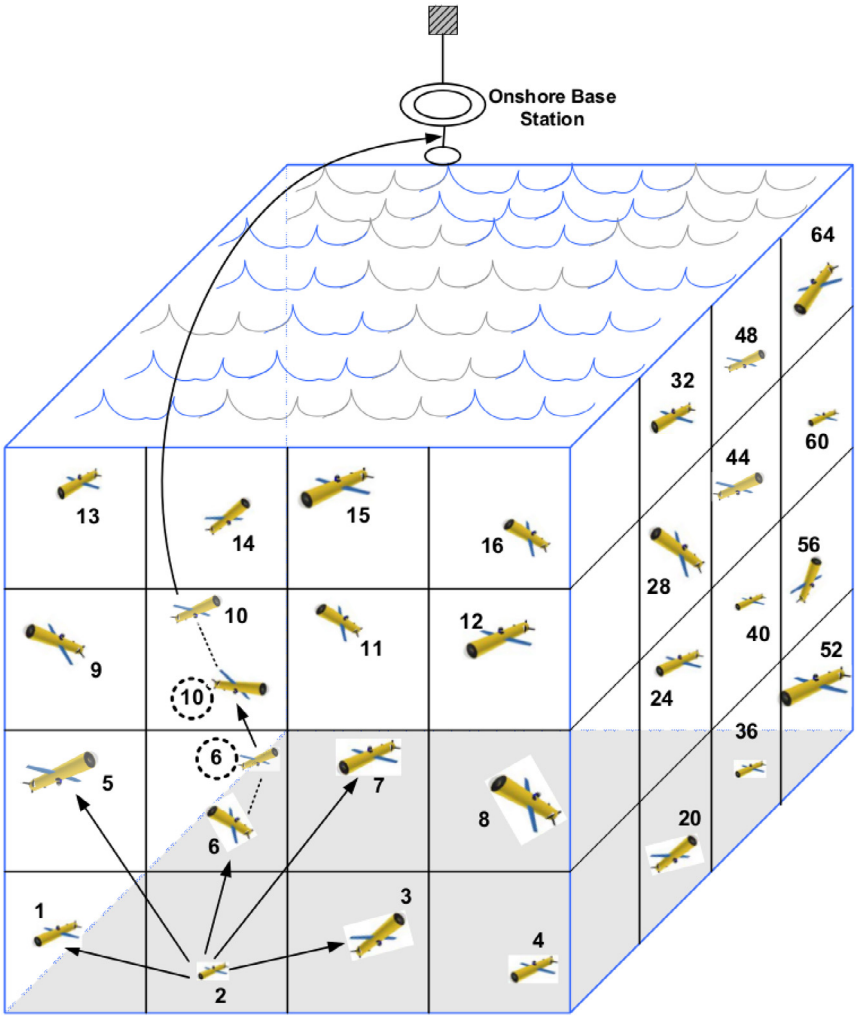

Fig. 1. An underwater sensor network based on AUV architecture. Node#6 and node#10 move closer so that node#10 can receive data from node#6. The dotted lines indicate node movement. Node#10 is the next scheduled surfacing node, called *target*.

its validity period. It is determined by the application. If a packet, after its creation, is not delivered before its lifetime expires, it is no longer valid. Therefore, if a packet is generated at time t_g, t_c can be calculated as,

$$t_c = t_g + T_d. \tag{1}$$

Consider a packet which is generated at 8:00 am and its T_d is 1 h, the packet is expected to reach the *target* by 9:00 am.

The goal of this work is to find a node within one or more hops which can deliver data within their given deadline. Let us suppose that there are 64 nodes in the network as shown in Fig. 1. They are scheduled to surface one after another every one hour during the day, i.e., T_i is 1 h. Further assume that node 2 has some data packets (intermediate reports) to send. It first searches for a *target* (10 as shown in the figure) which is scheduled to surface next. Node 2 then searches for a suitable next hop (say node 6) to forward data towards node 10. It calculates that node 6 is within its communication range. So it immediately forwards data to it. Upon receiving the data packets, node 6 discovers that node 10 is its one hop neighbor but not within its communication range. In order to be within 10's communication range, node 6 discovers that it along with node 10 has to move closer to each other. To accomplish this, node 6 gives an explicit command to node 10 instructing it to initiate movement and at the same time starts moving itself. Once both 6 and 10 are within each other's communication range, node 6 starts forwarding its stored data packets to node 10. After receiving, node 10 goes to surface and offloads those data to the onshore station.

ADDP relies on the capability of the AUVs to exchange information with one another using acoustic technology. A node maintains two communication ranges, (i) Control communication range (r_c) and (ii) Data communication range (r_d). Control packets are transmitted at lower data rates and hence can propagate over longer distance than data packets. Control communication range is set longer than data communication range so that nodes residing in neighboring areas can participate in control packet exchange regardless their position in the respective areas. For example, in Fig. 1, node 2 can exchange control packets with nodes 1, 3, 5, 6, and 7, irrespective of their position in the allocated areas. For data transmission, the protocol limits the communication range to save energy. In the rest of our discussion, we will use node and AUV interchangeably to represent an autonomous underwater mobile vehicle in ADDP.

4. Network initialization

During initialization, each AUV knows area map, $a = \{a(i) \mid 1 \leq i \leq n\}$, where $a(i)$ is the monitoring area of node i. A node performs sensing operation within its own area and is allowed to communicate with other nodes in its neighboring area. Each AUV also knows the boundary of its allocated area before deployment.

The location (x_i, y_i, z_i) of a node i is limited within its own predefined area $a(i)$. Having known the control channel communication range (r_c) from manufacturer's specification, the dimension of each cubic cell (d) is then initialized as $d = r_c/2\sqrt{3}$ and the data channel communication range r_d is set as $d/2$. This ensures that control channel covers the two farthest point between adjacent cells. Nodes are also assumed to know the surfacing schedule, $t_s = \{t_s(i) \mid 1 \leq i \leq n\}$, where $t_s(i)$ is the surfacing time of node i. An onshore station is aware of the deployment topology and initializes each AUV with a, t_s and T_d. AUVs are also considered to be synchronized throughout their operation. The surfacing schedule is maintained by using a circular queue. The first element of the queue is always the next node to surface. At every T_i, each AUV updates the first element and appends it to the last. All of the other elements are shifted to the left. The first element is updated by adding $n \times T_i$ to its current value. In this way, all AUVs maintain an updated information about the surfacing time of other nodes. The duration T_{unv} is intuitively assigned for all nodes in the network by the application user during deployment. Other important parameters are synchronization timers (e.g., request data transmission and data reception timers) used in our proposed scheme. They are not required to be initialized as their values are determined on the fly considering the status of network nodes and traffic.

5. Overview of ADDP

In ADDP, data packets sensed by mobile AUVs are expected to offload within t_c. The protocol enables AUVs to deliver data of a given underwater area using command-based controlled movement. Each AUV is assumed to calculate its own location. The protocol is based on a time-scheduled operation where each AUV finds a *target* node which meets t_c deadline and initiates forwarding of its stored data packets towards that node. In case of a tie, the protocol selects the node as a *target* that surfaces earlier.

In ADDP, a node can play the following three roles in the network, (i) source, (ii) forwarder, and (iii) *target*. Source nodes generate data packets, while forwarder nodes relay data packets until they reach the *target*. As mentioned before, nodes in the neighboring areas may not remain within each other's data communication range. In such cases, nodes are directed to move using commands through control packets. This makes both sender (source/forwarder) and receiver (forwarder/*target*) of the data packets to become available within each other's communication range so that a transmission can take place. A node is called a potential receiver and ready to receive data packets if it remains within the sender's data communication range. Once a node receives data packets from another node but is not a *target* of the packets, it becomes a sender itself and forwards the data packets towards the *target* node. Data packets are transmitted in a multi-hop fashion until the *target* node is reached.

ADDP works in four phases/functions: (i) find a *target* node as a destination of the data packets (details in Section 6.1); (ii) exchange control packets with the receiver to determine if forced movement is required (Section 6.2); (iii) command-based forced movement of AUVs (Section 6.3); and (iv) data transmissions to the next hop forwarder or the *target* (Section 6.4).

An overview of ADDP is shown in Fig. 2. After knowing a, t_c and t_s, each node goes to its dedicated area, moves within that area, generates/waits for data packets and performs operations according to the logical sequence of the functions represented in Fig. 2. In relation to the figure, the major functions of ADDP are detailed as follows:

- **Store data packets, and find a target and next hop forwarder**: Once an AUV generates its first data packet, a local search is made to find a *target* among the nodes in the network whose surfacing time and the time required to reach that *target* ensures t_c. If the generated packet is not a first packet, the node stores it into its local buffer. A node can also receive data packets from the other nodes in the network to forward to the *target*. After receiving data packets from other nodes, an AUV also searches for a *target* if delay experienced by incoming packets is higher than the packets stored in its local buffer. Otherwise, the AUV stores received data packets into its local buffer. Note, as alluded before, for either receiving its first own data packet or the delay experienced by incoming packets is higher than the stored packets, the node determines its next hop forwarder to reach the *target* and starts/updates a request timer.
- **Exchange control packets with next hop (Receiver)**: Once this timer expires, the sender exchanges control packets (REQUEST-REPLY) and checks whether both sender and receiver are within each other's data communication range.
- **Perform forced movement**: Forced movement is required if either of the sender or receiver or both are not within each other's communication range.
- **Forward data packets**: Once both of them come within their communication range, data packets stored at sender's buffer are transmitted to the receiver. At each hop, a local search is made to find a *target* node and a next hop forwarder. The process continues until data reach the *target*.
- **Go to surface, offload data and resume data sensing**: Once the *target*'s surfacing time expires, it goes back to surface, offloads data, recharges itself and returns to its assigned area.

The details of data transmission process is presented in the following section.

6. Data transmission in ADDP using mobile UASNs

To describe the data transmission process of ADDP, we have developed an algorithm as presented in Algorithm 1 . The variables used in the algorithm are initialized in Line 1. When a node generates its first data packet or receives packets from other nodes having higher delay than its stored packets, a modified *Breadth-First Search* (BFS) is called in Line 5 to determine a *target* and the next hop to reach that *target*. This is shown in Algorithm 2, the details of which is covered in Section 6.1. A node starts a request timer or updates an existing timer if the new timer expires before the old one. It then sends a "REQUEST" packet to the next hop forwarder/target when the timer expires which is given in Lines 7–10. As mentioned before, the communication range of control packets ("REQUEST" and "REPLY") is greater than that of the range of data packets. When a node receives a "REQUEST" packet and itself is a potential next hop forwarder or the *target* of the data packets, it sends a "REPLY" packet and participates in forced movement if needed. This is described in Lines 11–26. When a sender node receives a "REPLY" packet, it starts forced movement if the receiver is beyond its data communication range. Before initiating the forced movement, the sender starts a data transmission timer. Otherwise, if the sender finds that the receiver is within its data communication range, it starts data transmission immediately. This is described in Lines 27–38. After movement, the sender transmits data packets once its data transmission timer expires. This is shown in Lines 39–40. When the surface timer of a node expires, it goes to surface which is shown in Lines 41–42.

6.1. Finding the target using modified BFS

We have used a modified BFS to find the *target* node. The search is called if any of the following events occur:

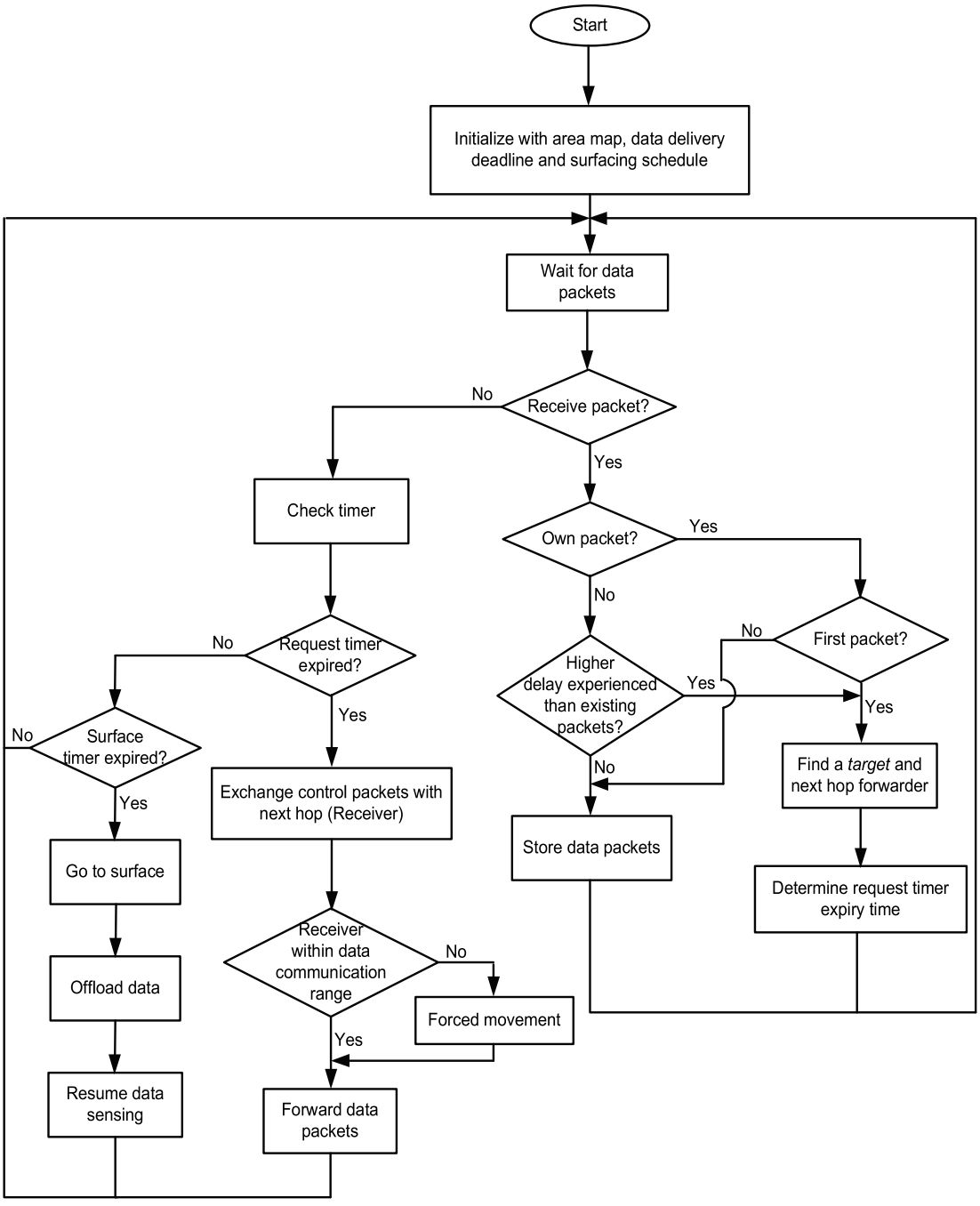

Fig. 2. An overview of AUV-based data delivery protocol (ADDP).

- Event 1: a node generates its first data packet,
- Event 2: a node receives data packets from other nodes in the network that experience larger delay than the delay experienced by packets in its local buffer.

In our modified BFS, a node u which meets the following two conditions is determined as a potential candidate to be a *target* by a node i if:

- Condition 1: $t_s(u) \leq current\_time + (T_d - max\_delay)$ and
- Condition 2: $t_s(u) - current\_time - h_c(u) \times (T_m + T_t(i) + T_p) > 0$.

Here, $t_s(u)$ and max_delay are the surfacing time of node u and the maximum delay experienced by a packet currently stored in node i's buffer, respectively; T_m is the maximum amount of time that it takes for the sender and receiver to come into each other's communication range; $T_t(i)$ is the amount of time required to transmit data packets stored in i's buffer; T_p is the maximum one-hop propagation delay to transmit data packets and $h_c(u)$ is the number of hops required to reach u from node i. Each node calculates T_m as,

$$T_m = \frac{d_{nb\_max} - r_d}{v} \tag{2}$$

where, d_{nb_max} is the maximum distance between any two points in neighboring areas and v is AUVs' velocity. The quantity $d_{nb\_max} - r_d$ indicates the maximum distance that the sender and receiver may need to move altogether to become available within each other's data communication range.

Algorithm 1: Data Transmission of node i in Mobile UASNs

Ensure: Data transmission to the selected next hop node
1: Initialize $target = next\_hop = null$
2: **if** node i receives data packets **then**
3: **if** (first data packet generated by the node itself) or (packet from other nodes with higher delay experienced than existing packets) **then**
4: Determine max_delay = The maximum delay of data packets stored in node i's buffer
5: Use Algorithm 2 to determine $target$ and $next\_hop$
6: **end if**
7: **else if** request timer of node i expires **then**
8: **if** $next\_hop \neq null$ **then**
9: Send "REQUEST" packet to the $next\_hop$ node
10: **end if**
11: **else if** node receives "REQUEST" packet **then**
12: **if** node is the next hop forwarder or $target$ (receiver) **then**
13: **if** sender is outside data communication range **then**
14: Determine sender's movement distance towards the receiver
15: **if** sender is outside data communication range after movement **then**
16: Determine movement distance of the receiver
17: Send "REPLY" packet to the sender and start moving
18: **else**
19: Send "REPLY" packet to the sender
20: **end if**
21: **else**
22: Send "REPLY" packet to the sender
23: **end if**
24: **else**
25: Do nothing
26: **end if**
27: **else if** node i receives "REPLY" packet **then**
28: **if** node i is the source or forwarder (sender) **then**
29: **if** receiver is outside data communication range **then**
30: Calculate own movement duration and that of the receiver according to Equation 14 and Equation 15
31: Schedule a data transmission timer with an expiry period T_{tr} according to Equation 16
32: Start moving
33: **else**
34: Transmit data packets to the next hop
35: **end if**
36: **else**
37: Do nothing
38: **end if**
39: **else if** data transmission timer expires **then**
40: Transmit data packets to the next hop
41: **else if** surface timer expires **then**
42: Go to surface and offload data
43: **end if**

Algorithm 2: Determine the $target$ and $next\_hop$ node for node i

Require:
1: n: Number of mobile nodes
2: t_s: Surfacing schedule of a set of mobile nodes, $t_s = \{t_s(j) | 1 \leq j \leq n\}$
3: $T_t(i)$: Data transmission duration of node i
4: T_m: Maximum node movement duration
5: T_p: Maximum one hop propagation delay
6: T_d: Data delivery deadline
7:
Ensure: An updated value of $target$ and the next hop to reach the $target$
8: Initialize $Q = \{\}$
9: **for** $j = 1$ to n **do**
10: $parent(j) = h_c(j) = null$
11: **end for**
12: Enqueue i to Q
13: **while** $Q \neq empty$ **do**
14: u = extract the element from Q
15: **if** $(t_s(u) \leq (current\ time + (T_d - max\_delay)))and((t_s(u) - current\ time - h_c(u) \times (T_m + T_t(i) + T_p)) > 0)$ **then**
16: **if** $parent(u) = null$ **then**
17: $target = u$
18: Break
19: **else**
20: **if** $target = null$ **then**
21: $target = u$
22: **else if** $parent(u) = parent(target)$ **then**
23: **if** $t_s(u) < t_s(target)$ **then**
24: $target = u$
25: **end if**
26: **else**
27: Break
28: **end if**
29: **end if**
30: **else**
31: **for** each neighbor j of u **do**
32: **if** $j \notin Q$ **then**
33: Enqueue j to Q
34: $h_c(j) = h_c(u) + 1$
35: $parent(j) = u$
36: **end if**
37: **end for**
38: **end if**
39: **end while**
40: Calculate request timer expiry time, $R_x = \frac{t_s(target) - current\ time - h_c(target) \times (T_m + T_t(i) + T_p)}{h_c(target)}$
41: $current\_destination = target$
42: **while** $parent(current\_destination) \neq i$ **do**
43: $current\_destination = parent(current\_destination)$
44: **end while**
45: Set $current\_destination$ as the $next\_hop$ forwarder of node i to reach the $target$
46: Start a request timer with R_x

The first condition determines whether the surfacing time of a potential candidate, u satisfies t_c for packets stored in node i's buffer. It also takes into account any delay already suffered by those packets after generation. The second condition ensures that $t_s(u)$ is long enough so that i's data packets reach u having maximum delay at each hop. Node i starts a request timer considering the amount of delay required at each hop in the worst case to reach a particular $target$. The delay at each hop includes: (i) T_m, (ii) $T_t(i)$ and (iii) T_p. Therefore, as shown in Lines 40 of Algorithm 2 , node i calculates a request timer according to the remaining time that it has to reach the $target$, i.e., $(t_s(u) - current\_time)$ and the time, i.e., $(h_c(u) \times (T_m + T_t(i) + T_p))$, node i requires further to transmit the stored packets. The timer is used to determine data hold time at node i's local buffer. A node is selected as a $target$ if data packets can reach that node within t_c considering T_m, T_p and $T_t(i)$ at each hop.

Each node calculates or updates max_delay (in Condition 1) each time it generates a new data packet or receives data packets from other nodes. When a node is the source of data packets stored in

its local buffer, the delay experienced by those packets is calculated as, font=

$$max\_delay = \begin{cases} 0 & if\, first\, packet \\ max\_delay + (current\_time \\ \quad -last\_update\_time) & other wise. \end{cases} \quad (3)$$

Here, *last_update_time* is the time when the last packet was inserted into the buffer.

Before receiving a bunch of data packets from other nodes, each node receives a control packet ("DATAHEAD") containing the maximum delay already suffered by those packets and is denoted as, *prev_max_delay*. The *max_delay*, therefore, can be updated as,

$$max\_delay = \begin{cases} prev\_max\_delay & if\ prev\_max\_delay \\ & > max\_delay \\ unchanged & other wise. \end{cases} \quad (4)$$

6.2. Exchange of control packets

The ADDP protocol allows packet accumulation at each hop and exchanges one control packet for a train of data packets. Since $r_c > r_d$, nodes at neighboring areas can participate in control packet exchange no matter whether they are within each other's data communication range. The transmission distance of control packets is set so that the farthest two points of two neighboring areas can be covered. As mentioned in Section 6.1, once a *target* and next hop forwarder is determined, an AUV starts a request timer with an expiry period (R_x) after which it starts exchanging control packets with its next hop. Once the timer expires, the node sends a "REQUEST" packet to the selected next hop. After receiving the "REQUEST" packet if an AUV finds itself as a potential next hop forwarder or *target*, it sends a "REPLY" packet to the sender of the "REQUEST" packet. By exchanging control packets, both sender and receiver determine whether they need to move forcefully closer to each other for transferring data packets. The details of forced movement technique are discussed in the next section.

6.3. Forced movement

As mentioned earlier, each node in ADDP moves within its predefined area. When a node (r) receives a "REQUEST" packet, it checks if its current distance to the sender (s), denoted as, *d_cur* is less than the data communication range, i.e., $d_{cur} < r_d$ as shown in Fig. 3. If it is, no forced movement is required. Otherwise, the receiver determines whether a forced movement of the sender or both the sender and receiver is needed. Receiver movement is required if sender's maximum movement towards the receiver does not bring their distance within r_d.

The receiver calculates its new distance to the sender (d_{mov_s}) considering sender's maximum movement towards itself. As shown in Fig. 4, the new location of the sender for its maximum movement is (x, y, z). To calculate d_{mov_s}, the receiver first calculates the point (x, y, z) where the sender is expected to move in order to reduce the distance between them. Let us assume that the receiver knows its own location (x_r, y_r, z_r) and also knows the initial location of the sender (x_s, y_s, z_s) from the "REQUEST" packet. The receiver draws an imaginary line between these two points as shown in the figure. The receiver also knows the boundary plane between their respective regions.

The boundary plane between two regions can be determined by the points (x_1, y_1, z_1), (x_2, y_2, z_2) and (x_3, y_3, z_3). The line joining two AUVs passing through the points (x_s, y_s, z_s) and (x_r, y_r, z_r) intersect the boundary plane at a point which is the maximum movement point or future location (x, y, z) of the sender towards the receiver and can be determined by the following equations that are detailed in [31],

$$\begin{vmatrix} x & y & z & 1 \\ x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \end{vmatrix} = 0 \quad (5)$$

$$x = x_s + (x_r - x_s) \times p \quad (6)$$

$$y = y_s + (y_r - y_s) \times p \quad (7)$$

$$z = z_s + (z_r - z_s) \times p. \quad (8)$$

The value of p can be calculated as,

$$p = \frac{\begin{vmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_s \\ y_1 & y_2 & y_3 & y_s \\ z_1 & z_2 & z_3 & z_s \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & (x_r - x_s) \\ y_1 & y_2 & y_3 & (y_r - y_s) \\ z_1 & z_2 & z_3 & (z_r - z_s) \end{vmatrix}}. \quad (9)$$

The value of p in Eq. (9) can then be re-used in Eqs. (6)–(8) to determine the point of intersection (x, y, z). The sender is expected to move to (x, y, z) as shown in Fig. 4. Therefore, using Eqs. (6)–(8), the receiver calculates distance (d_{mov_s}) between itself (x_r, y_r, z_r) and the sender's new location (x, y, z). If $d_{mov\_s} > r_d$, receiver's forced movement is required.

After the forced movement of the receiver, its future location (x', y', z') towards the sender can be represented by a set of parametric equations between points (x, y, z) and (x_r, y_r, z_r) as,

$$x' = x + \delta(x_r - x) \quad (10)$$

$$y' = y + \delta(y_r - y) \quad (11)$$

$$z' = z + \delta(z_r - z) \quad (12)$$

where, $0 \leq \delta \leq 1$ and δ can be calculated as,

$$\delta = \frac{r_d}{d_{mov\_s}}. \quad (13)$$

The value of δ is chosen in a way such that the new distance between sender and receiver remains within their data communication range, i.e., $d_{mov\_r} < r_d$. The new location (x', y', z') of the receiver (shown in Fig. 5) along the line joining (x, y, z) and (x_r, y_r, z_r) can be calculated by putting the value of δ into Eqs. (10), (11) and (12).

6.4. Data transmission to the next hop

Data forwarding takes place if both the sender and the receiver are within each other's data communication range. After receiving a "REPLY" packet, an AUV transmits data packets immediately if no command-based forced movement is required. Otherwise, the node calculates a data transmission timer and starts sending/forwarding data packets once the timer expires. Eq. (16) in Section 7.6 shows how the transmission timer is calculated.

6.5. Computational complexity of algorithms

If the neighbor table of n AUVs is represented by a graph, each AUV becomes a vertex and the links between an AUV and its neighboring AUVs are the edges of a graph. Each operation of Algorithm 2 takes $O(1)$ computational time. Since Algorithm 2 finds the target and next hop forwarder nodes using modified breadth first search algorithm, for m edges, in the worst case, the computational complexity of Algorithm 2 is $O(n + m)$ [32]. Depending on the distribution of AUVs, $O(m) \in [O(1), O(n^2)]$. Since each AUV is

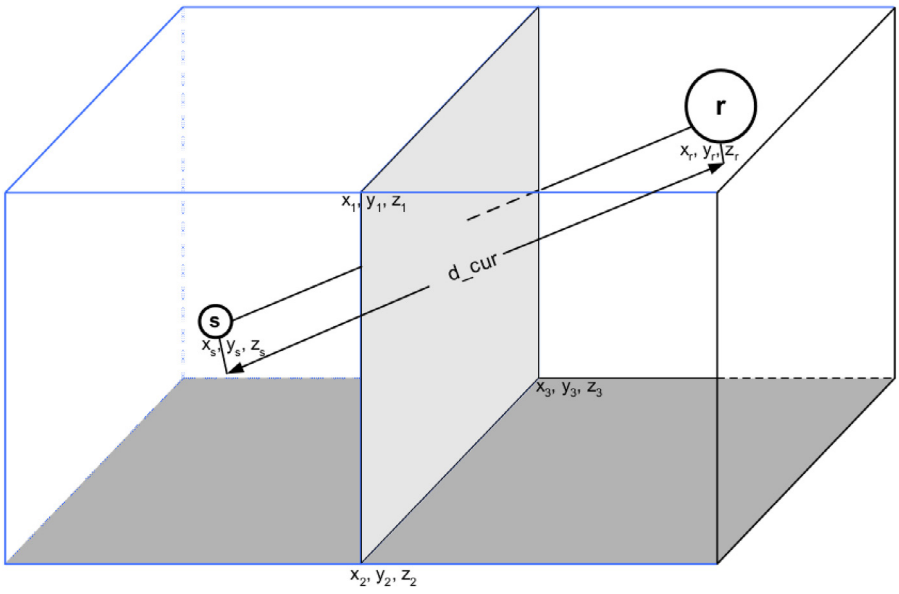

Fig. 3. Node location before movement of AUVs.

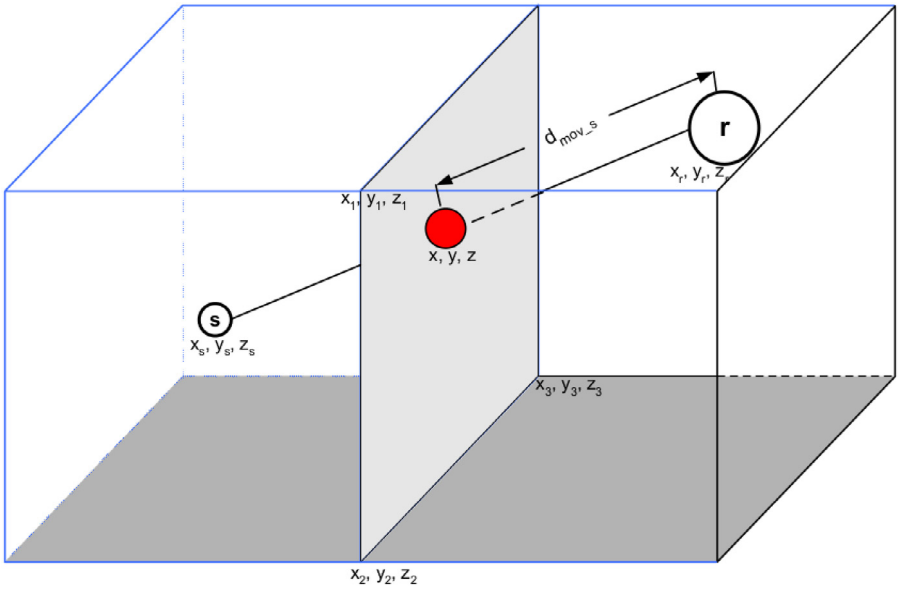

Fig. 4. Movement of the node sending data in ADDP.

deployed in an equal and particular cubic sub-area and its movement is limited to that sub-area, AUVs are uniformly distributed throughout the 3D deployment area. This implies $O(m) \rightarrow O(n)$. Therefore, the overall computational complexity of Algorithm 2 is $O(n)$.

Algorithm 1 uses Algorithm 2 to find the target and next hop forwarder nodes. Algorithm 1 also executes each operation except Algorithm 2 in $O(1)$ computational time. For this reason, the overall computational complexity of Algorithm 1 is the same as that of Algorithm 2 , i.e., $O(n)$.

7. State transition of nodes

During the execution of ADDP, a node can be in one of the ten states during its lifetime as shown in Fig. 6. The first state (IDLE) is the initial state for any AUV in the network. In this state, a node senses the environment, generates its own packets and receives packets from other nodes. The second state (SURFACE) shows that a node is ready to surface and offload its stored data packets. The next four states (REQ_SEND, REQ_RECV, REP_SEND, REP_RECV) indicate that the node is currently busy to exchange control packets (REQUEST-REPLY) with its next hop forwarder. The following two states (MOVE_SND and MOVE_RCV) show that the node is participating in forced movement and the final two states (DATA_SEND and DATA_RECV) depict that the node is busy in sending or receiving data packets.

The state transition diagram in Fig. 7 graphically depicts the operation of ADDP using a sequence of states that a node goes through during its life cycle. In order to avoid cluttering the diagram, self loops are not shown. A detailed description about each state and the event that triggers a node to move from one state to another is detailed in the following subsections.

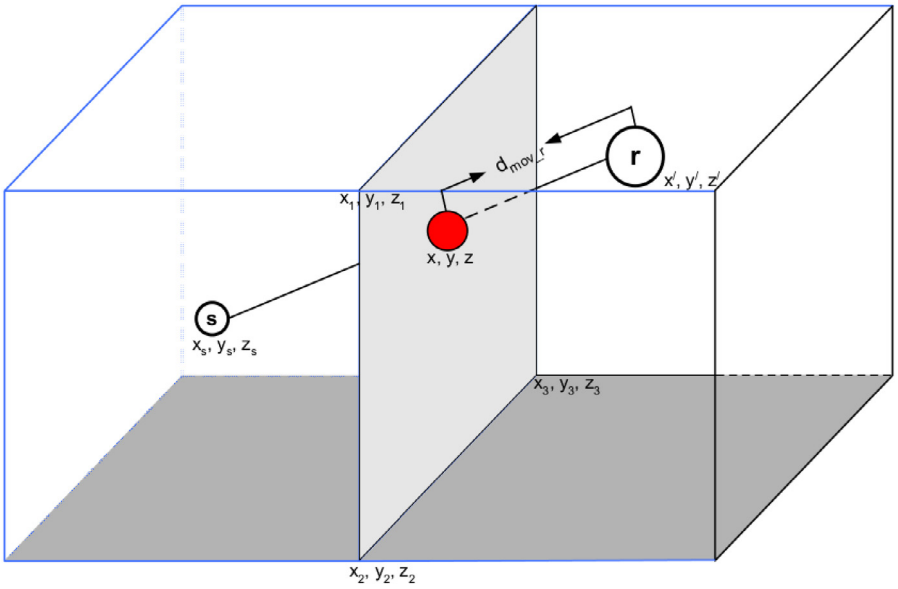

Fig. 5. Forced movement of the node receiving data in ADDP.

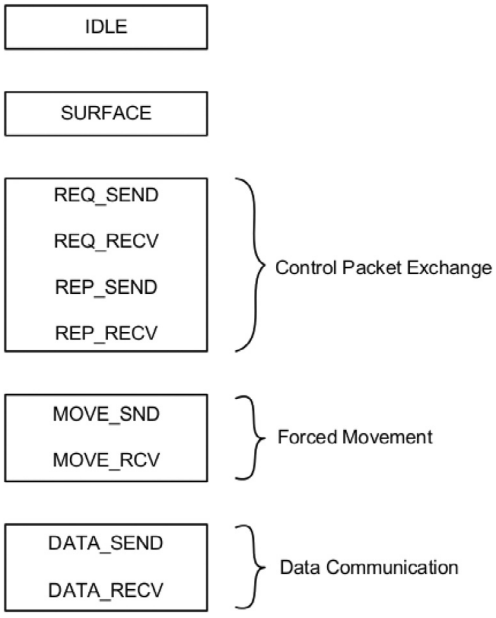

Fig. 6. AUV states.

7.1. IDLE state

- Node i initializes its surface timer $t_s(i)$ when it goes to its pre-defined area in water. If the surface timer expires, node i goes to the SURFACE state.
- A node executes a modified BFS as explained in Section 6.1 to find the *target* (destination node in ADDP) and starts a request timer. The request timer determines when a node should start exchanging control messages with its next hop forwarder. Once the timer expires, it sends a "REQUEST" packet to the next hop forwarder and switches to the REQ_SEND state.
- When a node is in IDLE state and receives a "REQUEST" packet, it goes to the REQ_RECV state.
- The data transmission timer determines when a node can send or forward its stored data packets after necessary forced movement (explained in Section 7.6). If no movement is required, a node determines to transmit data packets immediately. In both cases, when a node is ready to transmit its data packet while it is in the IDLE state, it moves to the DATA_SEND state to start data transmission.
- When a node receives data packets from the other nodes, while it is in the IDLE state, it moves to the DATA_RECV state.

7.2. SURFACE state

- When a node is in the SURFACE state, it offloads its stored data packets to the surface station and recharges itself.
- After offloading, the node goes back to its assigned area under water for collecting data packets and switches to the IDLE state.

7.3. REQ_SEND state

- If a node receives a "REQUEST" packet when it is in the REQ_SEND state, it simply discards the packet and remains in its current state.
- When a node is in the REQ_SEND state and receives a "REPLY" packet from its potential next hop forwarder, it moves to the REP_RECV state.

7.4. REQ_RECV state

- If a node receives "REQUEST" packet when it is in the REQ_RECV state, it simply discards the packet and remains in its current state.
- If a node is in the REQ_RECV state and the node itself is a valid next hop forwarder or *target* of the "REQUEST" packet, it sends a "REPLY" packet and goes to the REP_SEND state.
- If a node is not a valid next hop forwarder or *target* while it is in the REQ_RECV state, it simply discards the packet and goes to the IDLE state.

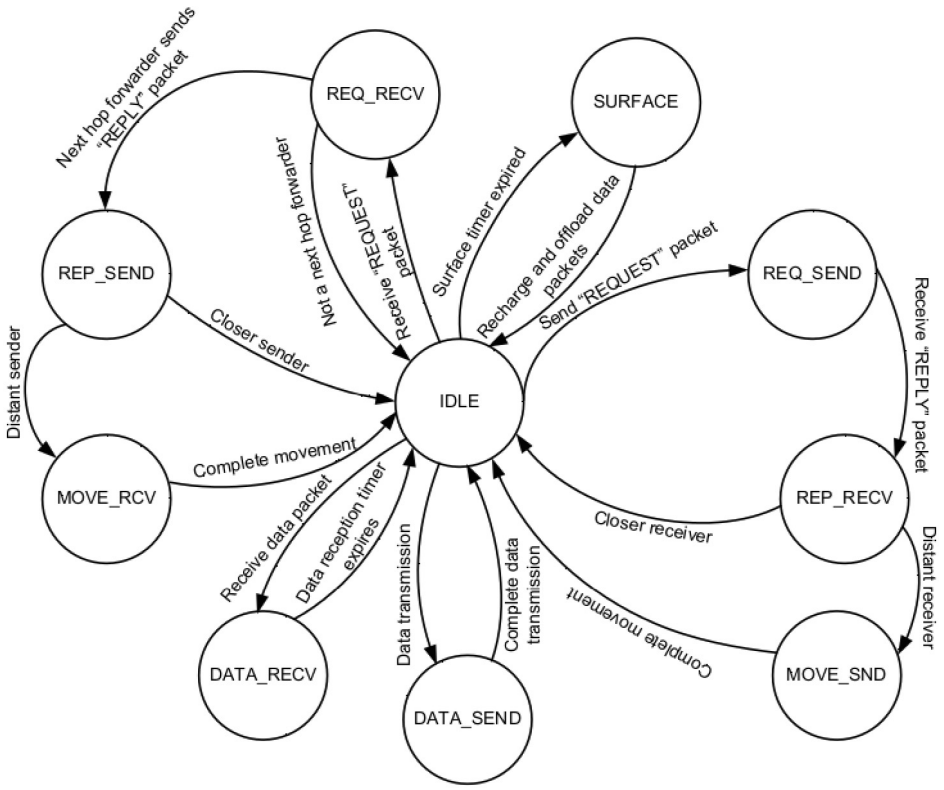

Fig. 7. ADDP operation.

7.5. REP_SEND state

- If a node is in the REP_SEND state and finds that movement of either sender or receiver is not required, i.e., $d_{cur} < r_d$ as shown in Fig. 3, it goes to the IDLE state.
- If a node finds that $d_{cur} > r_d$ while is it in the REP_SEND state, it calculates the distance between itself and sender's future location after maximum movement, i.e., d_{mov_s} (Fig. 4). If $d_{mov\_s} > r_d$, the movement of the receiver is required and the node goes to the MOVE_RCV state, otherwise it goes to the IDLE state.
- If a node is in the REP_SEND state and receives a "REQUEST" or "REPLY" packet, it simply discards the packet and remains in its current state.

7.6. REP_RECV state

- If a node is in the REP_RECV state and $d_{cur} < r_d$, (Fig. 3), i.e., both sender and receiver are within each other's data communication range, no movement is required. The node goes to the IDLE state.
- If a node is in the REP_RECV state and finds that $d_{cur} > r_d$, i.e., node movement is required, it calculates the movement duration of itself as,

$$move\_dur(s) = \frac{d((x_s, y_s, z_s), (x, y, z))}{v} \tag{14}$$

where, as mentioned before and shown in Fig. 4, (x_s, y_s, z_s) and (x, y, z) are current and future locations of the sender, respectively; v is its current velocity and $d((x_s, y_s, z_s), (x, y, z))$ represents the distance between (x_s, y_s, z_s) and (x, y, z). The sender also calculates the movement duration of the receiver as,

$$move\_dur(r) = \frac{(d(x_r, y_r, z_r), (x', y', z'))}{v} \tag{15}$$

where, (x_r, y_r, z_r) and (x', y', z') are current and future locations of the receiver, respectively.

If no movement is required, the sender starts data transmission immediately. Otherwise, the sender node starts a data transmission timer and moves to the MOVE_SND state. The data transmission timer is calculated as,

$$T_{tr} = max(move_{dur}(s), move_{dur}(r)). \tag{16}$$

7.7. MOVE_SND state

A sender node starts a forced movement while it is in the MOVE_SND state. At this state, a node stops its autonomous movement and starts forced movement. After completing movement, the sender moves to the IDLE state.

7.8. MOVE_RCV state

A receiver node starts a forced movement while it is in the MOVE_RCV state. A node switches from the MOVE_RCV state to the IDLE state once it completes its movement and reaches its new desired location.

7.9. DATA_SEND state

- When a node is in the DATA_SEND state, it will send its stored data packets to the next hop forwarder towards the *target* node.
- Once a node forwards all its data packets to the next hop, it switches back to the IDLE state.

7.10. DATA_RECV state

While a node remains in the DATA_RECV state, it buffers the packets received from the other nodes. In this state, the node starts a data reception timer which is calculated using the following equation as,

$$T_r = B_s \times T_{td} + T_{pd} \tag{17}$$

where, the quantity B_s is the number of packets the sender wants to send to the receiver, T_{td} is the transmission delay to transmit one data packets and T_{pd} is one hop propagation delay between the sender and receiver. When a node sends data packets to the receiver, the value of B_s is incorporated in a separate control packet ("DATAHEAD") and sent to the receiver followed by the data packets so that receiver can calculate T_r. After receiving, the receiver buffers all the successive data packets while remaining in the DATA_RECV state. After the timer in Eq. (17) expires, the receiving node switches back to the IDLE state and starts its autonomous movement.

8. Performance evaluation

In this section, we assess the performance of our proposed ADDP by conducting a set of experiments in NS2 [33]. To the best of our knowledge, there are no opportunistic communication protocols designed for fully AUV-based ad hoc underwater acoustic networks with which we can compare the performance of our protocol. Therefore, we have chosen DBMR [13] (a depth-based routing protocol) for the purpose of comparison and modified it to make it suitable for our fully AUV-based UASNs. However, the basic behavior of the protocol remains the same. Similar to ADDP, as DBMR supports node mobility, the nodes in DBMR are assigned to different areas and they can move within their own areas. Data delivered to sink nodes (known as *target* in ADDP) are offloaded to the surface at regular intervals.

We first set the simulation environment in such a way so that it can approximate the realistic underwater deployment scenario closely and then evaluate how various design parameters affect the network performance, such as packet delivery ratio, routing overhead and energy consumption detailed in Section 8.2. These parameters are studied by varying the traffic load, source density and network size for different scenarios.

8.1. Simulation environment

The experiments were conducted in a 3D multihop network topology comprising 5 to 30 mobile nodes (AUVs) to create different scenarios. Node density was considered to be one node per 500 × 500×500 m^3. The nodes were placed uniformly within the region of interest, i.e., the entire 3D region was divided into equal-sized cubic cells (volume of one cell = 500 m × 500 m × 500 m) and one node was placed randomly in each cell. The maximum data communication range was 0.5 times the size of a cell, i.e., 250 m omni-directional. The acoustic transmission power for data communication was set as 3 W. The characteristics of underwater acoustic channel were modeled applying the physical layer properties described in [34] in ns2 simulator. The channel data rate was set as 50 kbps.

Nodes were considered to follow a "Random Waypoint" mobility model which is widely used in mobile Ah hoc networks [35] and cellular networks [36], and the nodes move within their own predefined area at a velocity of 1–3 m/s [37,20]. The lifetime of a packet was considered to be 300 min and the nodes were assumed to surface at an interval of 100 min. Two nodes are called neighbors if they are located in neighboring cells. The two farthest points between neighboring cells are the two diagonally opposite corners of those cells. The maximum distance between these two points in adjacent cells is $2\sqrt{3}d$, where d = 500 m is the width of each cell and the cells are diagonally adjacent to each other. In the experiments, the values of the ADDP specific parameters such as maximum one hop transmission range and channel data rate for exchanging control packets were set as $2\sqrt{3}d$ and 10 kbps, respectively. The simulation parameters are listed in Table 1.

Table 1
Simulation settings.

Volume of each AUV deployment	500 × 500 × 500 m^3
Mobility model	Random waypoint
Node velocity	1–3 m/s
Channel data rate	10–50 kbps
Data channel communication range	250 m
AUV surfacing interval	100 min

8.2. Performance metrics

The metrics used to analyze the performance of ADDP protocol are as follows:

- **Packet Delivery Ratio (PDR)**: The ratio between the number of packets successfully received by the targets and those sent by the source nodes. It indicates the reliability of the protocol under different network conditions.
- **Routing Overhead**: A portion of the total traffic that is expended as control messages. This metric measures the efficiency of the protocol in expending control overhead to deliver data packets.
- **Energy Consumption**: The total energy spent for transmission and reception of data and control packets. This metric is used to measure the energy efficiency of ADDP protocol.

Each simulation ran for 45 h for each protocol and for both protocols similar scenarios were used. The results of 20 simulation runs are averaged and reported in the following subsections. The performance of ADDP is compared with DBMR in terms of PDR, routing overhead and energy consumption under varying network load, source density and network size. We performed paired t−test comparing the two protocols and in all metrics the performance improvement by PRADD is statistically significant at a 99% confidence level.

8.3. Simulation results and analysis

8.3.1. Effects of network load

The performance of ADDP was evaluated and compared with DBMR by varying the packet generation interval. To analyze the effects of network load, the data packet generation interval per source was varied from 1 to 10 s. For this experiment, the network size was set to 20 nodes and the number of sources was 4. The source nodes were chosen randomly to generate data packets.

Fig. 8 illustrates the PDR of ADDP and DBMR with varying packet generation interval. Both protocols suffer in the region of high traffic load mainly due to collision as shown in the left side of the figure. As the network gets slower (low traffic load), the PDR increases for both protocols due to less number of collisions. However, ADDP achieves higher PDR (e.g., 0.69–0.93) than that of DBMR (e.g., 0.43–0.69) irrespective of data generation rate. The forced movement mechanism (described in Section 6.3) allows ADDP to communicate with nodes that are not initially within each other's data communication range. Therefore, our proposed protocol achieves a moderate level of reliability (above 68%) for the highest network loads, reaching an acceptable high delivery rate

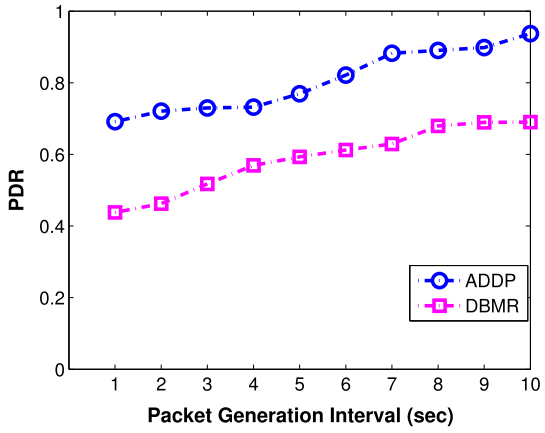

Fig. 8. Comparison of data delivery success between ADDP and DBMR protocols with varying traffic.

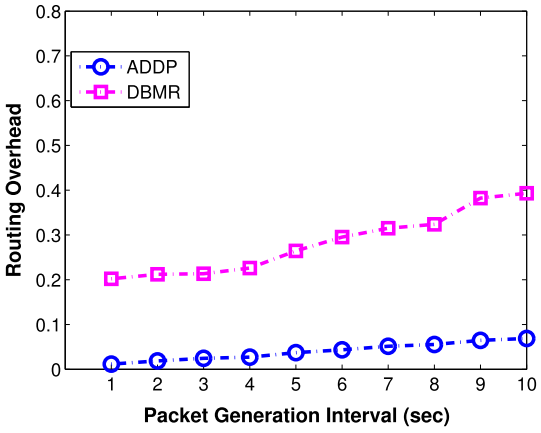

Fig. 9. Routing overhead vs. packet generation interval in both protocols.

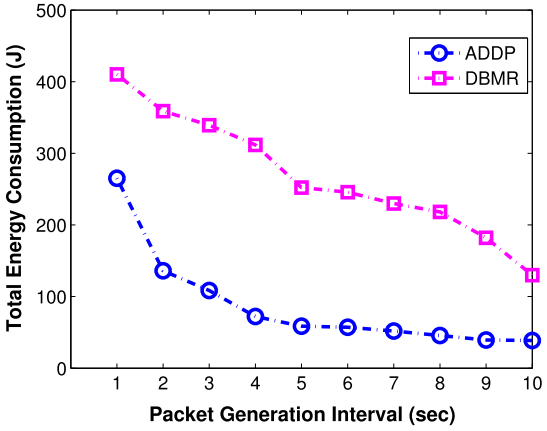

Fig. 10. Comparison of energy consumption in both protocols with varying packet generation interval.

(93%) at lower data generation. The PDR of DBMR falls (0.46 at 2 s interval) due to its next hop selection mechanism which is based on depth value. As the height of each cubic AUV cell is twice the data communication range, DBMR cannot find a valid next hop (lower depth value) within the sender's data communication range. Even when a next hop node is found, it could move out of range between two consecutive data generations giving a false impression about its presence. When a next hop node moves out of the sender's data communication range, transmission drops which incurs packet loss, and consequently reduces PDR in this protocol. On the other hand, ADDP allows the sender to move over its maximum distance towards the receiver so that it can transmit data packets even if they are not initially within each other's communication range. In cases where sender movement alone is not sufficient enough to bring it within the receiver's communication range, the movement of receiver is employed in our protocol. Therefore, ADDP ensures data delivery regardless of the location of the next hop in the neighboring areas, resulting improved PDR.

Fig. 9 shows that the routing overhead of both ADDP and DBMR increases with decreasing network load. The less number of data generations increases the overhead per packet at low traffic load. Overall, the overhead in ADDP is significantly lower than that of DBMR. The route discovery process incurs more routing overhead in DBMR than our protocol. The reason is that DBMR does not take any measure into account for void region problem which becomes more severe at lower traffic load. Moreover, it triggers route discovery when the next hop is not found or moves out due to mobility, which increases its overhead. On the other hand, ADDP reduces routing overhead by transmitting a stream of data packets for one exchange of control packet (REQUEST-REPLY). The protocol employs a modified BFS to find a *target* node with the shortest hop count. It reduces overhead as control message exchange takes place at each hop and is proportional to hop count. In addition, the forced movement technique improves PDR in ADDP which results in reduced routing overhead as more successful transmissions take place with less number of control packets exchanges.

Fig. 10 demonstrates the energy consumption of both ADDP and DBMR with respect to packet generation interval. As the traffic load decreases (high packet generation interval), the energy consumption decreases. However, the decreasing rate in DBMR is lower than ADDP as seen in Fig. 10. This is due to the higher PDR (see Fig. 8) and less routing overhead (see Fig. 9) achieved by ADDP compared with those for DBMR. The reduced routing overhead along with next hop forwarder's availability during data transmission (by incorporating forced movement) contributes to the saving of energy in ADDP.

8.3.2. *Effects of source density*

Some underwater applications require data delivery protocols to deal with different number of sources. Increasing the number of sources introduces traffic flow from different points in the network. Routing protocols need to cope with this situation efficiently. For this experiment, the network size was set to 20 nodes and the number of sources was varied from 1 to 10 nodes. Data sources were chosen randomly to generate data at a rate of 1 packet/s.

In Fig. 11, the performance of both ADDP and DBMR is evaluated in terms of PDR varying the number of data sources. Both protocols show a decreasing trend in PDR as the number of active sources increases. But the decreasing rate is considerably lower in our protocol than DBMR. The main reason for higher decline in the PDR of DBMR is due to its inefficient route discovery technique. As seen in the graph, our protocol achieves 95% of PDR when only one source generates data packets and maintains a stable PDR (above 0.56) throughout the entire range. This is due to its efficient selection of *target* node and utilization of node movement at each hop to ensure data delivery. The explanation for this improved PDR is similar to that given for the PDR vs. packet generation interval graph.

Fig. 12 shows that the routing overhead increases only by a modest amount in ADDP with increasing number of data sources. Though the overhead rises at higher node density for both protocols, the rate of overhead increase in ADDP is still significantly lower than DBMR. The increased traffic load, due to the increasing number of sources, increases the chance of packet collisions in

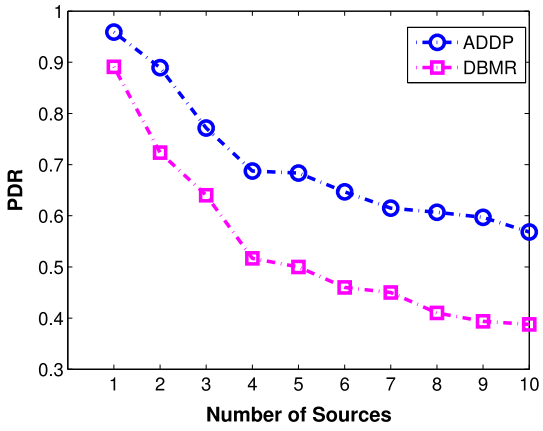

Fig. 11. Variation of packet delivery success with the number of data generation sources.

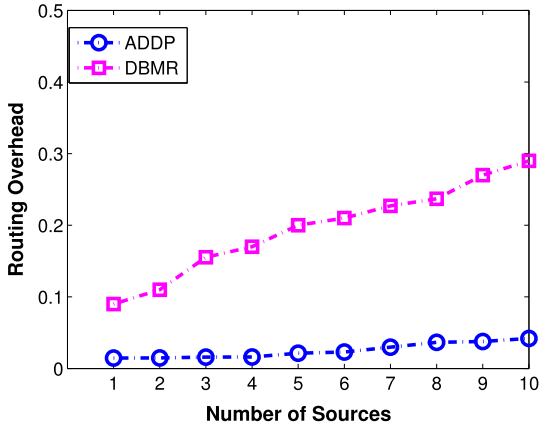

Fig. 12. Routing overhead vs. the number of data generation sources.

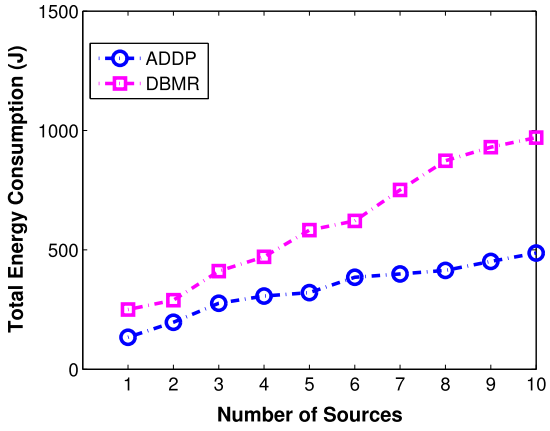

Fig. 13. Energy consumption with varying number of data generation sources.

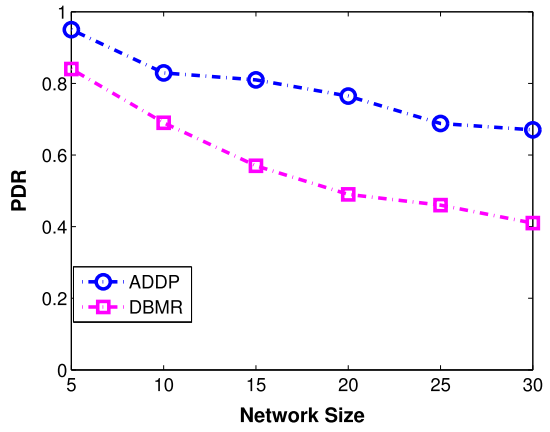

Fig. 14. Effect of network size on data delivery success.

the network which in turn, increases the overhead as less and less number of packets are successfully delivered. However, due to collision and void region problem, in DBMR, a source node needs to attempt next hop discovery a number of times before becoming successful which increases its overhead. Moreover, additional control packet exchanges are required to find the additional routing paths for data delivery which also contributes to DBMR's routing overhead. It is evident from the figure that our protocol's overhead increases by a small amount compared to DBMR as the data generation sources increase. This is due to the same factors that reduce its overhead in Fig. 9. Moreover, the protocol has a higher communication range for exchanging control packets which enables communication between neighboring nodes regardless of their positions in their respective cells. This increases the chance of neighbor's availability, which in turn, prevents repeated exchanges of control packets and reduces the overall routing overhead. ADDP also considers node movement time and accumulates data packets as much as possible before t_c expires. This serves two purposes: (i) improves PDR as data reach the *target* within t_c and (ii) reduces overhead as more data packets are transmitted with the less number of control packet exchanges.

Fig. 13 shows that the energy consumption increases for both ADDP and DBMR as the number of data sources increases, which in turn, increases traffic load. But the increase is significantly higher in DBMR than our protocol. With higher number of sources, DBMR requires more frequent route discovery which increases its overhead and energy consumption. On the other hand, ADDP adds benefits in data transmission by transmitting a large number of data packets with little control overhead. As a consequence, ADDP is energy efficient.

8.3.3. *Effects of network size*

This experiment was carried out to investigate the performance of the protocols by adding more nodes to the network, i.e., evaluating the scalability of the protocol. The network size was varied from 5 to 30 nodes and 20% nodes were randomly selected to act as data sources. Data packets were generated 1 packet/s.

Fig. 14 plots the PDR of both protocols for different network sizes and shows that ADDP is superior to DBMR in all cases. Our protocol maintains a PDR of more than 65% throughout the entire network range. For small network size (5 to 15 nodes), ADDP achieves above 80% PDR. The figure also indicates that ADDP scales better than DBMR as the network size increases. The PDR drops below 50% for DBMR when the network size reaches 20, while the PDR of ADDP still maintains a respectable level of PDR (0.76). When the network size increases, the number of active sources grows causing an increase in traffic load which increases the chance of packet collision and decreases the PDR. Moreover, in larger networks, packets have to travel more hops in order to reach the destination node. The also increases the chance of collisions at each hop and reduces the PDR. The decline in PDR in DBMR is due to its inability to recover from void regions and find a next hop outside its data communication range. The forced movement technique enables transmitting data reliably where DBMR fails to deliver.

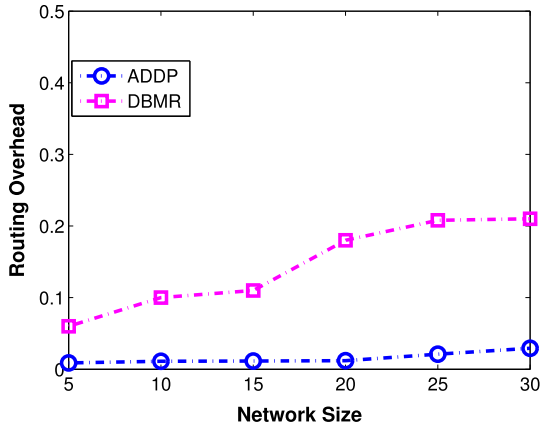

Fig. 15. Comparison of routing overhead in both protocols with growing network size.

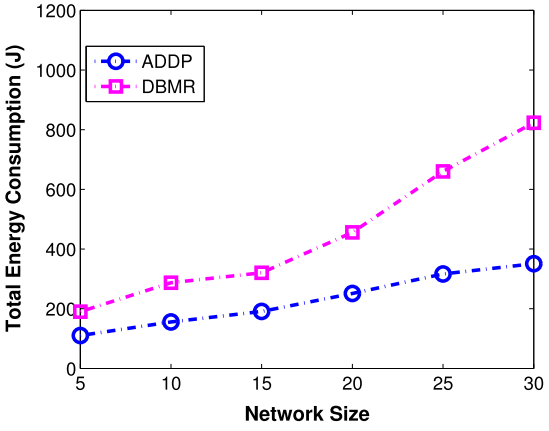

Fig. 16. Energy consumption vs. network size in both protocols.

Fig. 15 shows that the routing overhead of ADDP remains reasonably low and stable throughout the experiment. The performance of DBMR degrades drastically with network size due to high number of collisions in large network. This causes the ratio of control traffic to data traffic to increase for both protocols. DBMR suffers most in terms of overhead due to its inefficient route discovery technique because of the same reasons detailed in Fig. 9. The increased overhead is further accumulated at each hop as the network size grows. On the other hand, ADDP incurs less overhead than DBMR for all cases and is scalable in response to increased network size.

The energy consumption of both ADDP and DBMR increases with the increase in network size as shown in Fig. 16. However, energy consumption increases at a much slower rate in our protocol than DBMR which indicates that ADDP is more energy efficient for all network sizes. For medium (20 nodes) to large network size (30 nodes), the energy consumed for transmission and reception of control and data packets rises sharply for DBMR due its frequent route discovery. It wastes energy by sending data packets to a forwarder which may have moved out of reach. This incurs more routing overhead, and consequently, increases the energy consumption in DBMR. However, as mentioned before, ADDP transmits a bunch of packets when a next hop is available which, along with its efficient *target* selection and node movement technique, reduces the overall energy consumption compared to DBMR.

9. Conclusion

In this paper, we have shown the suitability and benefit of applying AUVs to forward sensed data from underwater to the surface in an ad hoc manner. We have identified the necessity to employ AUV-based architecture to support mission critical applications. The protocol designed in this paper finds AUVs which can deliver the data packets within their given deadline. The protocol employs a modified BFS technique to find a *target* node as a destination of the data packets and a next hop forwarder to reach that *target*. Control packets are exchanged with the forwarder to determine whether forced movement is required. After necessary movements, data transmission to the next hop forwarder or the *target* takes place once both the sender and receiver of the packet are within each other's data communication range. The experimental results presented in this paper prove the merit of the proposed technique, demonstrate how it helps in improving data delivery with low overhead and exhibit good scalability.

References

[1] X. Che, I. Wells, G. Dickers, P. Kear, X. Gong, Re-evaluation of RF electromagnetic communication in underwater sensor networks, IEEE Commun. Mag. 48 (12) (2010) 143–151.
[2] P. Xie, J.-H. Cui, L. Lao, VBF: vector-based forwarding protocol for underwater sensor networks, in: Proceedings of the 5th International IFIP-TC6 Conference on Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems (NETWORKING'06), 2006, pp. 1216–1221, http://dx.doi.org/10.1007/11753810_111.
[3] N. Nicolaou, A. SEE, P. Xie, J.-H. Cui, D. Maggiorini, Improving the robustness of location-based routing for underwater sensor networks, in: Proceedings of OCEANS 2007 - Europe, 2007, pp. 1–6, http://dx.doi.org/10.1109/OCEANSE.2007.4302470.
[4] D. Shin, D. Hwang, D. Kim, DFR: An efficient directional flooding-based routing protocol in underwater sensor networks, Wirel. Commun. Mob. Comput. 12 (17) (2012) 1517–1527. http://dx.doi.org/10.1002/wcm.1079.
[5] H. Yan, Z.J. Shi, J.-H. Cui, DBR: Depth-based routing for underwater sensor networks, in: Proceedings of the 7th International IFIP-TC6 Networking Conference on AdHoc and Sensor Networks, Wireless Networks, Next Generation Internet (NETWORKING'08), 2008, pp. 72–86.
[6] Z. Li, Y. Nianmin, Q. Gao, Relative distance based forwarding protocol for underwater wireless networks, Int. J. Distrib. Sens. Netw. 2014 (173089) (2014) 11. http://dx.doi.org/10.1155/2014/173089.
[7] A. Amory, T. Tosik, E. Maehle, A Load balancing behavior for underwater robot swarms to increase mission time and fault tolerance, in: Proceedings of the 2014 IEEE International Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014, pp. 1306–1313, http://dx.doi.org/10.1109/IPDPSW.2014.146.
[8] S. Karthik, Underwater vehicle for surveillance with navigation and swarm network communication, Indian J. Sci. Technol. 7 (S6) (2014) 22–31.
[9] F. Aznar, M. Sempere, M. Pujol, R. Rizo, J. Pujol, Modelling oil-spill detection with swarm drones, Abstr. Appl. Anal. 2014 (949407) (2014) 14.
[10] J. Wang, W. Shi, L. Xu, L. Zhou, Q. Niu, et al., Design of optical-acoustic hybrid underwater wireless sensor network, J. Netw. Comput. Appl. 92 (2017) 59–67.
[11] S.C. Dhongdi, P. Nahar, R. Sethunathan, L.J. Gudino, K. Anupama, Cross-layer protocol stack development for three-dimensional underwater Acoustic Sensor Network, J. Netw. Comput. Appl. 92 (2017) 3–19.
[12] L. Liu, P. Wang, R. Wang, Propagation control of data forwarding in opportunistic underwater sensor networks, Comput. Netw. 114 (2017) 80–94.
[13] U.D. Prasan, G. Mahapatra, Energy efficient multiple sink variation to the depth based route protocol for underwater sensor network, Int. J. Eng. Sci. Adv. Technol. 2 (4) (2012) 951–954.
[14] U. Lee, P. Wang, Y. Noh, F. Vieira, M. Gerla, J.-H. Cui, Pressure routing for underwater sensor networks, in: Proceedings of IEEE INFOCOM, 2010, pp. 1–9, http://dx.doi.org/10.1109/INFCOM.2010.5461986.
[15] A. Wahid, D. Kim, An energy efficient localization-free routing protocol for underwater wireless sensor networks, Int. J. Distrib. Sens. Netw. 2012 (307246) (2012) 1–12. http://dx.doi.org/10.1155/2012/307246.
[16] M.C. Domingo, A distributed energy-aware routing protocol for underwater wireless sensor networks, Wirel. Pers. Commun. 57 (4) (2011) 607–627. http://dx.doi.org/10.1007/s11277-009-9864-3.
[17] W.-G. Seah, H.-X. Tan, Multipath virtual sink architecture for underwater sensor networks, in: Proceedings of IEEE OCEANS 2006 - Asia Pacific, 2006, pp. 1–6, http://doi.org/10.1109/OCEANSAP.2006.4393933.

[18] E. Magistretti, J. Kong, U. Lee, M. Gerla, P. Bellavista, A. Corradi, A mobile delay-tolerant approach to long-term energy-efficient underwater sensor networking, in: Proceedings of IEEE Wireless Communications and Networking Conference (WCNC), 2007, pp. 2866–2871, http://doi.org/10.1109/WCNC.2007.531.
[19] Z. Guo, G. Colombo, B. Wang, J.-H. Cui, D. Maggiorini, G. Rossi, Adaptive routing in underwater delay/disruption tolerant sensor networks, in: Proceedings of the Fifth Annual Conference on Wireless on Demand Network Systems and Services (WONS), 2008, pp. 31–39, http://doi.org/10.1109/WONS.2008.4459352.
[20] Z. Zhou, Z. Peng, J.-H. Cui, Z. Shi, A. Bagtzoglou, Scalable localization with mobility prediction for underwater sensor networks, IEEE Trans. Mob. Comput. 10 (3) (2011) 335–348. http://dx.doi.org/10.1109/TMC.2010.158.
[21] Z. Zhou, J.-H. Cui, S. Zhou, Efficient localization for large-scale underwater sensor networks, Ad Hoc Networks 8 (3) (2010) 267–279. http://dx.doi.org/10.1016/j.adhoc.2009.08.005.
[22] A.P. Das, S.M. Thampi, Fault-resilient localization for underwater sensor networks, Ad Hoc Networks 55 (2017) 132–142.
[23] E. Mortazavi, R. Javidan, M.J. Dehghani, V. Kavoosi, A robust method for underwater wireless sensor joint localization and synchronization, Ocean Eng. 137 (2017) 276–286.
[24] J. Zhang, Y. Han, C. Zheng, D. Sun, Underwater target localization using long baseline positioning system, Appl. Acoust. 111 (2016) 129–134.
[25] J.M. Jornet, M. Stojanovic, M. Zorzi, Focused beam routing protocol for underwater acoustic networks, in: Proceedings of the Third ACM International Workshop on Underwater Networks (WuWNeT), 2008, pp. 75–82, http://doi.org/10.1145/1410107.1410121.
[26] R.M. Gomathi, J.M. LeoManickam, T. Madhukumar, Energy preserved mobicast routing protocol with static node for underwater acoustic sensor network, in: Proceedings of the 2nd International Conference on Innovation Information in Computing Technologies(ICIICT), 2015, pp. 1–8.
[27] N. Nowsheen, G. Karmakar, J. Kamruzzaman, PRADD: A path reliability-aware data delivery protocol for underwater acoustic sensor networks, J. Netw. Comput. Appl. 75 (2016) 385–397.
[28] J. McMahon, E. Plaku, Autonomous data collection with limited time for underwater vehicles, IEEE Robotics and Automation Letters 2 (1) (2016) 112–119.
[29] A. Caiti, Cooperative behaviours of AUV teams and networked underwater communication: how to ask the way and not go astray, in: Proceedings of the 2nd International Conference on Underwater Networks & Systems, 2014.
[30] H. Zheng, J. Wu, Data collection and event detection in the deep sea with delay minimization, in: Proceedings of the 2nd IEEE International Conference on Sensing, Communication, and Networking (SECON), 2015, pp. 354–362.
[31] Line-Plane intersection. URL http://mathworld.wolfram.com/Line-PlaneIntersection.html.
[32] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Section 22.2: Breadth-first search, MIT Press and McGraw-Hill, 2001.
[33] The network simulator - ns2. URL http://www.isi.edu/nsnam/ns/.
[34] A.F. Harris III, M. Zorzi, Modeling the underwater acoustic channel in Ns2, in: Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools (ValueTools), 2007, pp. 18:1–18:8.
[35] S. Ahmed, G.C. Karmakar, J. Kamruzzaman, An environment-aware mobility model for wireless ad hoc network, Comput. Netw. 54 (9) (2010) 1470–1489.
[36] A. Haider, I. Gondal, J. Kamruzzaman, Dynamic dwell timer for hybrid vertical handover in 4G coupled networks, in: 73rd IEEE Vehicular Technology Conference (VTC Spring) 2011, pp. 1–5.
[37] J. Jaffe, C. Schurgers, Sensor networks of freely drifting autonomous underwater explorers, in: Proceedings of the 1st ACM International Workshop on Underwater Networks (WUWNet), 2006, pp. 93–96, http://doi.org/10.1145/1161039.1161058.

Gour Karmakar received the B.Sc. Eng. degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology in 1993 and Masters and Ph.D. degrees in Information Technology from the Faculty of Information Technology, Monash University, in 1999 and 2003, respectively. He is currently a senior lecturer at the School of Engineering and Information Technology, Faculty of Science and Technology, Federation University Australia. He has published more than 105 peer-reviewed research publications including twenty one international reputed journal papers and received an ARC linkage project grant. He has successfully supervised nine Ph.D. and three Masters by research students. His research interest includes multimedia signal processing, wireless sensor and social networks, Internet of things, simulation modeling and artificial intelligence. He is a member of IEEE.

Joarder Kamruzzaman received the B.Sc. and M.Sc. degrees in Electrical and Electronic engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, and the Ph.D. degree in Information Systems Engineering from Muroran Institute of Technology, Hokkaido, Japan. Currently, he is an Associate Professor in the Faculty of Science and Technology, Federation University Australia and an Adjunct Associate Professor in the Faculty of Information technology Monash University, Australia. His research interest includes wireless communications, sensor networks and computational intelligence. Kamruzzaman has published over 210 peer-reviewed publications which include 60 journal papers and edited two reference books on computational intelligence theory and applications. He is the recipient of Best Paper award in the IEEE WCNC'10Sydney, Australia; IEEE-ICNNSP'03, Nanjing, China; and ICICS'15, Singapore. His research has been supported by grants from industry and government, including prestigious Australian Research Council grant. He is currently serving as a program committee member of a number of international conferences and an Editor of the Journal of Network and Computer Applications.

Nusrat Nowsheen received her Ph.D. from Monash University Australia in 2015. Her research interest includes under water communications and wireless sensor networks.